DOCUMENT RESUME

ED 318 390 IR 014 218

AUTHOR Kotze, Paula
TITLE Implementing Computer-Aided Instruction in Distance Education: An Infrastructure. RR/89-06.
INSTITUTION South Africa Univ., Pretoria.
PUB DATE Aug 89
NOTE 40p.
PUB TYPE Reports - Research/Technical (143)

EDRS PRICE MF01/PC02 Plus Postage.
DESCRIPTORS *Computer Assisted Instruction; *Courseware; *Distance Education; Foreign Countries; Higher Education; *Individualized Instruction; *Instructional Development; Models; *Program Implementation
IDENTIFIERS *University of South Africa

ABSTRACT

The infrastructure required for the implementation of computer aided instruction is described with particular reference to the distance education environment at the University of South Africa. A review of the state of the art of online distance education in the United States and Europe is followed by an outline of the proposed infrastructure for implementing computer aided instruction at the University of South Africa. There are seven major components in this infrastructure, which is derived from a general model followed at the University of Hagen: (1) an author and supervisor/mentor of a specific courseware package; (2) the author system for developing the courseware; (3) a course database for storing the courseware that is ready for distribution to the students; (4) a "network" for distributing the courseware, either via an electronic medium or via the postal system; (5) the tutor system on which students run their courseware, which is attached to a communication support where applicable; (6) a laboratory containing computer software material, other than courseware, for doing practical work; and (7) the student. Each of these components and the interrelationships among them are described in detail with the aid of flowcharts and diagrams. (27 references) (GL)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

ED318390

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)
☒ This document has been reproduced as received from the person or organization originating it.
☐ Minor changes have been made to improve reproduction quality.
• Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

IMPLEMENTING COMPUTER-AIDED INSTRUCTION IN DISTANCE EDUCATION: AN INFRASTRUCTURE

BY
MRS. PAULA KOTZÉ

RR/89-06

Research Report
Department of Computer Science
and Information Systems
UNISA

Navorsingsverslag
Departement Rekenaarwetenskap
en Inligtingstelsels
UNISA

RR/89-06. 38 p. Pretoria, Augustus 1989.

P.O. BOX 392
PRETORIA
0001
SOUTH AFRICA

IR014218

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY
P. Kotze

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

BEST COPY AVAILABLE

IMPLEMENTING COMPUTER-AIDED INSTRUCTION IN DISTANCE EDUCATION: AN INFRASTRUCTURE

BY

PAULA KOTZÉ

PREFACE

Implementing Computer-Aided Instruction in distance education is like walking into a mine field of a war zone. So many things have to be kept in mind. Making one step into the wrong direction can be fatal. Every aspect of Computer-Aided Instruction has to be studied carefully before implementing it. This article will look at one of many such aspects - a possible infrastructure for implementing Computer-Aided Instruction in distance education.

CONTENTS

		Page
	Preface	(ii)
1.	INTRODUCTION	1
1.1	Current method of tuition	2
1.2	The new method of tuition	3
1.3	Scope of the paper	5
1.4	Some definitions	5
2.	STATE OF THE ART	6
3.	THE GENERAL CONCEPT - A PROPOSAL	8
3.1	Author/supervisor/mentor	9
3.2	Author system	16
3.2.1	The special needs of the audience	16
3.2.2	Cost considerations	16
3.2.3	Identification and definition of realistic training needs	17
3.2.4	CAI tools	17
3.2.4.1	Definition and design methodology	17
3.2.4.2	Hardware requirements	17
3.2.4.3	Software requirements	20
3.3	Tutor system	26
3.4	Communication support	28
3.5	Summary	30
4.	Conclusion	31
	BIBLIOGRAPHY	33

IMPLEMENTING COMPUTER-AIDED INSTRUCTION IN DISTANCE EDUCATION: AN INFRASTRUCTURE

MRS. PAULA KOTZÉ
Department of Computer Science and Information Systems
UNISA

1. INTRODUCTION

The University of South Africa, popularly referred to as Unisa, is the only university in South Africa which offers remote education. Unisa is the largest distance teaching institution in Africa - tuition is offered in six faculties (Arts, Science, Economic and Management Sciences, Law, Education and Theology), comprising 56 departments and more than 150 different subjects. Student numbers exceeded 100 000 for the first time during 1988. The current student number, as on Unisa census day (6 June), is 108372.

Being part of Africa, many of our students live in a Third World environment, whilst others have access to all the modern communication media of the western world. As a result, careful consideration has to be given to the choice of media used in computer-aided instruction at Unisa.

The area of computer-aided instruction is therefore being researched at present, from a number of different perspectives, by members of the Department of Computer Science and Information Systems, amongst others. This paper will cover some of these perspectives.

1.1. CURRENT METHOD OF TUITION

As is the case throughout Unisa, teaching in the Department of Computer Science and Information Systems is largely by means of the printed/written word. On registering, students receive study guides and tutorial letters, in which subject matter is explained and examples given. In most subjects students are required to buy a prescribed book, while the Unisa Library offers students a wide variety of additional reference material and periodicals.

The means of interaction that exist between students and their lecturers are mainly threefold - (rather compact) discussion classes, which take the form of lectures or workshops, are held at the regional offices of Unisa from time to time; the student may contact the lecturer personally either by visiting him in his office or contacting him by phone; or the student may correspond with the lecturer and address queries to him in letter form, which are answered individually by the lecturer.

Only recently a fourth means of communication between lecturers and students has emerged, and that is via electronic mail - together with a vast number of communication requirements and problems.

Students registering for programming modules in Computer Science (which are in fact compulsory modules for a degree or diploma in Computer Science or Information Systems) are currently required to have access to a personal computer, preferably IBM-PC compatible running under the DOS operating system. Students who do not have access to a private computer can make use of computers in the computer laboratories of Unisa at the main campus in Pretoria and at the regional offices. The computer time and facilities available in the laboratories are limited, however, and advance bookings have to be made.

Although many of our students have their own computer at home or have access to terminals or computers (at their employers, a friend or the Unisa laboratory), there is not yet a general realization of the gains to be made from using these computers for learning, training, experimenting and

communicating in the course of distance education. The computer is currently only used as a tool (word processing, statistical analysis, etc.) and/or a tutee (programming, designing, etc.), and not as a tutor.

In contrast to the situation at conventional universities, our students are faced with several specific problems owing to their remoteness from the university:

- The communication among students and between students and the university is rather limited.
- Most students can use their computers and course-related software for only a few weeks a year.
- During the past few years, the number of students in Computer Science and Information Systems at Unisa has increased at an unforeseeable rate (even though access to a computer has become compulsory). This has resulted in an even poorer ratio of number of students to teaching staff, and individual tutoring has become almost impossible. In addition it is hard, if not almost impossible, to get qualified and experienced teaching staff.

1.2 THE NEW METHOD OF TUITION

Since the current methods of tuition at Unisa offer little, if any, opportunity for interactive communication between students and the lecturers, there is a growing need for methods or approaches that will change or even merely improve the situation.

To overcome some of the above-mentioned problems, the department of Computer Science and Information Systems at Unisa has started researching the possibility of developing courseware based on electronic media for distance learning and education. Since students in other departments at Unisa are not required to have access to a computer, the Department of Computer Science and Information Systems is the only department in which computer-aided instruction can be implemented to any great extent.

When deciding on using computer-aided instruction, there are many factors that have to be considered before working out how the system can be implemented. Since there are no departments in Unisa that specialize in designing or programming courseware for computer-aided instruction, the Department of Computer Science and Information Systems has to do all of this itself. It is therefore important to make the implementation as easy as possible by choosing the most appropriate programming/authoring language or system, the most suitable hardware configuration and communication network (if possible), suiting the requirements of the subject and minimizing the administrative problems associated with a distance teaching university.

The main goals of the project are therefore to develop an attractive program for teaching computer science, together with a minimization of administrative tasks to be performed by the teaching staff - leaving them more time to improve the standard of tutoring in their specific subject area.

The salient features of this program are as follows:

- The student can work at home, in his own time, at an electronic station (personal computer(PC));
- the student's work-station (PC) is connected to a "communication network";
- the courseware (or some of it) is of an electronic form;
- the electronic courses are presented in an attractive and motivating form;
- the idea of "learning by doing" is achieved (by consequently integrating electronic laboratory sessions into courses, where possible);
- the courses will cover actual computer science topics and not only exercises or tests.

1.3 THE SCOPE OF THE PAPER

This paper will only cover the infrastructure needed for implementing CAI, with particular reference to distance education, and not the design and development of courseware as such. The actual design and development of courseware is a complete study on its own.

1.4 SOME DEFINITIONS

Distance learning can be considered as covering "the various forms of study at all levels which are not under the continuous, immediate supervision of tutors present with their students in lecture rooms or on the same premises, but which, nevertheless, benefit from the planning, guidance and tuition of a tutorial organisation" (Holmberg, 1978 in Davies & Preece 1988:75).

Courseware consists of material designed to help the student learn - it normally consists at least of a teacher's guide, student notes and a computer program.

The **modes of instructional applications** of computers are:

- Tutorials - here information is presented, replies are sought, and feedback given to the student. The computer assumes the total responsibility of instruction.
- Drill-and-Practice - examples are spelling practice and maths drills. In this mode no instruction is taking place. The teacher assumes total responsibility of instruction.
- Instructional gaming - the students applies certain skills or concepts in the game environment.
- Simulation - a simulation of some real situation is presented, i.e. the use of models to attain the essence of a system without having to use the real thing.
- Problem solving - the student may use an appropriate algorithm or develop his own in order to solve a problem with a computer.

- Instructional Management - student scheduling and grade averaging are good examples. Here the instructor rather than the student is the user of the program.
- Instructional Support - examples of this mode are inventory control, text generation, and text editing.
- Testing - this type of educational software can be used for interactive testing, for grading and analysing the tests of students, and for computing statistics such as item analysis and class averages.

(Price 1982:5,7)

2. THE STATE OF THE ART

Whilst we have not yet decided on a final configuration at Unisa a study of relevant literature has revealed that several similar research projects are currently under way in Europe and North America.

Schofield (1989:94) states that on-line education is an idea whose time is yet to come, but is certainly on the way. Distance learning is dear to the heart of the European Commission(EC), which has a big project called Delta, devoted to it. On-line study is part of it.

In the US:

- There is an Electronic University, based in San Francisco, which offers a complete MBA course taught entirely using computer communications. Students at many other US universities use on-line computing facilities as part of their studies (Schofield 1989:94).

- The New York Institute of technology, Westbury, New York, combines independent study with on-line communications. A student can obtain a baccalaureate degree in a distance learning mode using the computer as a communication device between the instructors and the student. The courses are still dependent on printed material but students join classmates in conferences based on specific courses.

- The University of Delaware, Newmark, Delaware, has been using Plato terminals off-campus since 1982 to provide lessons in mathematics, English and science to students who have been found to be deficient in those areas (Charp 1988:161-162).

The major research on this topic that would suit the Unisa situation has up to now came from institutions in Europe:

- In the United Kingdom (UK), the pioneering work has been done, as one would have expected, by the Open University (UKOU) with its DT200 course, run for the first time during 1988. The UKOU is the largest university in the UK and has considerable experience in teaching undergraduates by means of distance learning methods. In the past students doing a programming, or technology course had to do all their practical work using a terminal attached to one of two DEC 20's by the telephone lines. The terminals are situated in the study centres (Davies & Preece, 1988). The DT200 course is on "An introduction to Information Technology: Social and Technological Issues". The DT200 course involves participation in a conferencing system, using Cosy software developed at the University of Waterloo, Canada. Open University students use a PC-compatible from home (Schofield 1989:94).

- A similar project named "Tele-Education" is underway in the Netherlands. In 1984 the Dutch Open University of the Netherlands, together with Philips (manufacturer of electronic devices), Kluwer (publishing house), and PTT (national telephone services) started a group in order to find out in what way a computer could be used as a home educative terminal. The first students should have received their adaptive courseware in January 1987 (Boon & De Wolf 1988:193-195).

- The Department of Mathematics and Computer Science, University of Hagen (FernUniversität), West Germany, partly supported by the German Government, is working on a project to develop attractive "electronic courses" for continued education. The "FernUniversität Hagen" is the only university in Germany which offers remote education for several subjects, e.g. computer science. Their students may enrol for a full diploma degree in computer science, for a certificate covering a

specific area of computer science, or even for individual courses. The students mainly live at a distance from the university and, at the time, are taught in a similar way as the current method used at Unisa (Schlageter, Six, Stern, & Unger, 1988:9-10). Their problems are therefore similar to those of the average Unisa student.

- In co-operation with the Institutes for Information Processing (IIG) of the Technical University, Schiesstattgasse, Graz, Austria, the University of Karlsruhe launched a major CAI project in the summer of 1985. The project is based on the system AUTOOL, developed at the Technical University of Graz, an extension of the PLATO system adapted to the videotex system and the international CEPT standard. The research done on the CLEAR centres at the University of Texas at Dallas, USA, is done in collaboration with the project at IIG (Maurer & Makedon, 1988:107; Makedon & Maurer, 1988:93; Ottman: 1988:219).

3. THE GENERAL CONCEPT - A PROPOSAL

The infrastructure that is proposed for implementing CAI in the Unisa situation, is derived from the general model followed at the University of Hagen. Several changes have, however, been made to suit the specific requirements of the Unisa students. The infrastructure consists of the following major components (their relationship to one another are depicted in Figure 1):

- An author and supervisor/mentor of a specific courseware package.
- The author system for developing the courseware.
- A course data base for storing the courseware that is ready for distribution to students.
- A "network" for distributing such courseware, either via an electronic medium or via the postal system (diskettes).
- The tutor system on which students run their courseware, attached to a communication support, where applicable.
- A laboratory containing computer software material, other than courseware, used for doing practical work.
- The student sitting at his own computer, in his own time, studying by means of CAI.

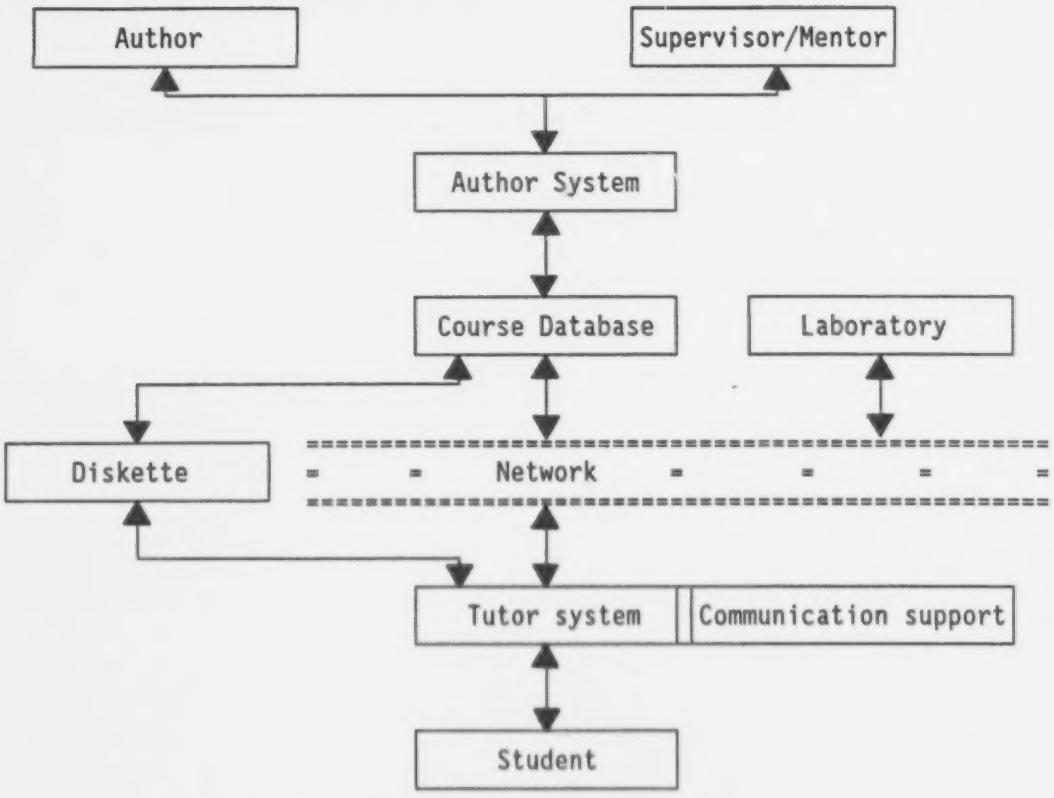


Figure 1: The general concept of the teaching system

The nature and functions of the different modules within a system such as that given in Figure 1 are as follows:

3.1 AUTHOR/SUPERVISOR/MENTOR

The design and development of CAI courseware is something of an enigma. Unlike more familiar media such as print, overheads, slides or video, what comes out bears no resemblance to what goes in - screen displays do not usually look like the program that generates them. Even for someone who knows how to program, instructional software (i.e. courseware) is perplexing because it is not enough that the program should merely run properly - it must also teach.

The requirement that courseware be instructionally effective as well as computationally correct, adds a new element to software development which is often missed by those who approach CBI from a computer science or data processing perspective. On the other hand, the development of courseware is inherently a programming task - a point sometimes overlooked by educators. Whilst a majority of people in this country can read and write, only very few will become successful authors. Similarly, computer programmers do not necessarily make successful CAI authors. Although the computer program is very important in a CAI lesson, much more is involved than just coding in a programming language. This involves careful specification of objectives, selection of programming strategies, detailed analysis of content structure sequence, development of pretests and post-tests, preliminary drafts, revisions, trials, validation and documentation (Kearsley 1985:209; Gleason 1981:11 Kotzé 1988:13-18).

A careful study of the special characteristics of CAI courseware authors reveals that three main areas of knowledge are required:

- Knowledge of content to be taught,
- knowledge of instructional systems design, and
- knowledge of computer programming (Burke 1982:12,37).

The development and production of **quality** CAI courseware is both a difficult and a time consuming task. Thus the successful CAI courseware author should have not only a thorough knowledge of the three areas mentioned above, but he should also be able to devote a lot of time to the planning, developing, and testing of the CAI courseware. The best candidate for writing quality software for education has teaching experience, knows both instructional systems design and computer programming well and has **plenty of enthusiasm** and all the **free time** he needs to spend on developing such software (Kontos 1985:16). Experienced CAI software developers use between 200 and 300 hours to develop one hour of delivered high quality conventional CAI software; intelligent-based CAI is thought to require one order of magnitude more (Betlach 1986:44).

A lecturer may know the content to be taught in the CAI lesson, but most likely does not know how to program the computer; a computer programmer knows how to program the computer but may know little, if anything, about the content to be taught in the CAI lesson. A teacher can learn computer programming and a computer programmer can research the content of the CAI lesson. As result, the two most widely used methods of preparing the CAI sequence have been, and still are as follows:

(1) Educators learning computer programming on their own, (just enough to write CAI lessons), and
(2) computer professionals who, on their own, learn about the educational process, (just enough to write CAI lessons).

These two methods of producing CAI lessons are very simplistic and have not produced, and are not likely to produce, more than just simple and unfortunately redundant CAI lessons (Kontos 1985:17; Kontos 1984:21).

There is no easy way that programmers can learn everything that CAI authors need to know in order to write good CAI courseware; neither is there an easy way in which educators can really learn the art of programming in good style and structure without effort. Extensive training of the teacher in the art of computer programming or of the computer professional in the art of teaching is absolutely necessary (Kontos 1985:17).

This brings us to the following two methods of preparing the CAI sequence:

(3) Computer professionals going through extensive and careful training to learn all the details of the educational process before they are allowed to apply it in the writing of CAI lessons, and
(4) educators going through extensive training in the art of computer programming in order to develop the ability to write good CAI lessons.

Of all the four methods mentioned above, the last one is the most cost effective and probably the most viable. A lot has been said about the teacher-author. Many so called experts in the field advocate that educators who master the art of using the computer to write CAI programs are the ones

likely to produce good CAI courseware at a minimal cost. But how easy or how difficult is it to master the art of using the computer for CAI development? How much training does the average teacher need in order to write the badly needed CAI? Is he capable or even willing to devote his time for such a difficult task? Who will fund such an extensive and costly training? Training the average teacher to write worthwhile CAI is not only time consuming but also very costly. Furthermore, it is doubtful that such an investment will eventually pay off. And even if the teacher knows programming, can he find the free time to work with the computer? It is well known that teaching by itself is a difficult, creative and time consuming job. Many educational institutions, especially universities and some colleges, in addition to heavy teaching loads, require their instructors to do research, publish regularly, attend professional meetings, get involved in curriculum development and other academic activities. Thus the real problem the teacher-author faces is time. The other problem is the lack of teacher-education courses in the two very important areas of instructional systems design and computer programming, tailored for the needs of teachers to help them in the development of quality CAI courseware. (Kontos 1985:17; Kontos 1984:21).

All four methods that we have examined this far have their problems. But there is other possibilities:

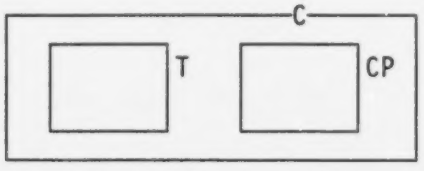
(5) Computer professionals and educators working together in a combined effort through productive communication and scientifically sound training with the objective of producing the best in CAI courseware.

This last method is not as simple to implement as it may appear to be. It has several variations which give birth to different models. Among these models, one will be singled out which is easier to implement and at the same time is likely to succeed in creating high quality CAI courseware.

Model A:

The team consists of both teachers(one or more) and computer professionals (one or more) working independently or in groups. Although a group in the team is made up of either teachers only or computer professionals only, there is some interaction and communication between groups and individuals. However, we may consider this model as two or more less separate and outonomous units, of teachers and other computer professionals (Kontos 1985:18).

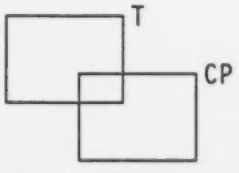
Model B:



As model A, but teachers and computer professionals in the team are under the guidance and supervision of a coordinator (Kontos 1985:18).

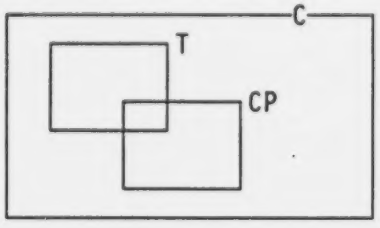
Models A and B can be thought of as extensions of methods (3) and (4) above, and they were included here also for the same reason.

Model C:



Teachers and computer professionals are separate units but they share resources (knowledge, information, materials, etc.). This model utilize the important feature of productive communication between educators and computer professionals within the team (Kontos 1985:18).

Model D:



As model C, but teachers and computer professionals in the team are under the guidance and supervision of a coordinator, a highly skilled individual forming the communication link between different teams which are engaged in the development and production of CAI courseware (Kontos 1985:18).

The Proposed Model CAI Team

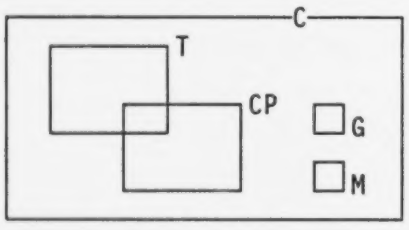


Figure 2: The model CAI team

The model for the development and production of CAI courseware (in both distance and conventional education) that is proposed is a team of professionals from both disciplines (education and computer science/technology). It consists of:

- Educators (at least one lecture/teacher) who are involved in all phases of the CAI development and production process but mostly in the design phase (T in Figure 2), sharing resources (knowledge, information, materials, etc.) with
- computer professionals (at least one programmer/computer specialist) again involved in all phases of the CAI process but mainly in the preparation, testing, etc. of the software (CP in Figure 2);
- a graphics artist (G) (optional);
- a co-ordinator (c) who is a highly skilled individual with years of experience in both software development and teaching; and
- a moderator (M) who is responsible for the documentation of a session (a meeting of all team members where project progress and problems are discussed).

This seems to be the most promising method and one that we should pursue at any cost. Only if both the programmers' and the educators' level of consciousness and sense of CAI becomes much more sophisticated, and only if they both agree that the gap between them should be bridged, will miracles happen in the CAI courseware development.

Models illustrating the team effort of this method for the development and production of CAI materials are currently in operation at several technologically advanced countries - for example in England, there is the CIC(Computers In the Curriculum) model, the "Five Ways" model, the ITMA model, and the "Buckshot" model (Watson 1983; Coates 1982).

This model (as described in Figure 2), however, is not without problems:

- One problem is deciding on the location from which the CAI team operates - will it be from a certain teaching department, from the computer centre, from a specially established buro for CAI, from outside the academic institution, etc.
- Another problem is the psychological battle that may arise between the two groups (teachers and programmers) as they become more competent in the major field of the other team. Some teachers will eventually become competent programmers, and some programmers will learn enough about education. One or more members of a team will want to prove that he/she can write and market a CAI lesson which he produced individually. This should be avoided as it will be very harmful to the very existence of the team and most important to the student population.
- Finally, one should realise that no matter what method or model is chosen, the spectre of redundancy of courseware always threatens one. Redundancy can be minimised if we know what has been done and what needs to be done in CAI lessons in this country. The only solution, I believe, is to have a central bank where all worthwhile CAI software of South Africa can been stored after being carefully selected and be readily available for reviewing by anyone who is interested.

3.2 AUTHOR SYSTEM

The author(s) and the mentor of a course use the author system to develop, edit and structure an electronic course. The author system also supports testing of a course (student's view) (Schlageter et. al. 1988:11).

In approaching any CAI methodology, attention may be centred upon several critical factors:

- the special needs of the audience;
- cost considerations;
- identification and definition of realistic training needs;
- CAI tools - definition and design methodology,
  - hardware requirements (system),
  - software requirements (language and database),
  - general instruction strategies for consideration in actual courseware instruction (Betlach 1986:35).

3.2.1 The special needs of the audience

At tertiary level, CAI should keep the needs and abilities of the adult learner in mind. Both levelled progression and regression provisions are essential to respond adequately to a diverse target training audience. Direct feedback and reinforcement mechanisms are just as essential. If CAI is to be successful, participants must view the CAI courseware as a non-threatening aid. CAI must provide an environment in which failure is not judged, but in which a productive learning experience can occur. The experience should be challenging and it should be fun.

3.2.2 Cost considerations

Both increased student interest and motivation are intrinsic to the CAI approach. In two sample studies, described by Pressman & Rosenbloom (1978:200-204), the decreased student drop-out rate was reflected as a direct result of increased student interest and motivation. CAI can respond directly to individualized participant needs through its capacity for individualized instruction, feedback and pacing. It therefore becomes very attractive in terms of time saved and overall payback analysis. In fact, CAI is a long term investment yielding real returns.

User payback, in the form of qualitative and quantitative results, addresses cost factors directly. But CAI costs relative to its unique components must also be considered: Hardware, software, user training, and maintenance and installation (Pressman & Rosenbloom 1978:194-195).

Beyond the normal quantitative spectrum, realistic cost analysis must address such factors as increased motivation, topic comprehension, active participation and involvement, etc. These key factors do not easily translate into economic benefits but often take some time to become discernible within the organisation (Betlach 1986:41).

3.2.3 Identification and definition of realistic training needs

From the academic instructor's point of view, key considerations in CAI authorship include more than just coding in a programming language: As said before, CAI preparation must involve careful specification of objectives, selection of programming strategies, detailed analysis of content structure sequence, development of pretests and post-tests, preliminary drafts, revisions, trials, validation, and documentation. (Kearsley 1985:209; Gleason 1981:11; Kotzé 1988:13-18). This regimen requires extensive training of the authors prior to CAI authoring.

3.2.4 CAI tools

3.2.4.1 Definition and design methodology

As with any other computerized system, in order to be successful the CAI courseware design should follow some kind of design methodology. The discussion of such design methodologies is, however, beyond the scope of this paper.

3.2.4.2 Hardware requirements

When the UKOU set up its DT200 course, virtually all the press interest was in the micro the students would use. Almost no attention was given to where the course materials come from (Schofield 1989:94).

The purpose of the system selection process is to specify a computer system that will perform one or more specific functions for a price that is worth it.

Although we are tackling the aspect of hardware first, the evaluation of hardware and software must happen simultaneously, unless one already possesses one or the other, in which case there is no choice.

There are two major types of commercial CAI systems available: "Dedicated" and "piggyback". Dedicated computer-based instruction (CBI) systems are developed and used exclusively for training purposes and involve special terminals and software designed specifically for instructional applications. Piggyback systems are software packages which run on existing hardware, allowing CAI capabilities to be added to a computer system already in use for some other purpose. The central question is this: What do we intend using the computer for, and what are our future plans? (Coburn, Kelman, Roberts, Snyder, Watt & Weiner 1982:87; Betlach 1986:43).

General rules for buying hardware are at least as tricky as those for buying software. Nonetheless a few general rules can be presented - although following them will not always guarantee success.

- Look for local sales and service.
- Beware of new products - almost all hot new products go through a teething phase, and will the vendor that you buy from today be in business tomorrow?
- Look for durable construction and engineering - if students are going to use it, it has to be rugged.
- Allow for expansion - will the vendor of the hardware provide either a bridge that extends the capabilities of my equipment or another computer to which I can readily move.
- Look for good and well-documented systems software.
- Choose a system that allows the use of more than one computer language.
- Look for a system with a lot of available applications software.
- Talk to owners and users of the systems you are considering.

- Don't rush anything.
- Don't try to do too much with too little computing power.

(Perry 1986:51-56; Coburn, et. al. 1982:92-94,107)

For CAI there are a few explicit questions to be asked when considering hardware:

- What hardware is required for developing lessons?
- What hardware is required for running completed lessons?
- Are graphics capabilities required?
  - Would an IBM-PC or compatible with a Hercules card be adequate, or do you need a CGA or even an EGA?
  - Would a monochrome graphics display screen be suitable, or is a colour graphics screen compulsory?
- Will the computers be linked up in a network? What additional hardware is required to do so?
- Under which operating system does the computer run - DOS, Macintosh, Unix, OS/2 etc.?
- Will students buy their own delivery systems, or must a computer laboratory be set up?
- Are qualified staff available for the installation and maintenance of such equipment?

Unfortunately, on-line education requires that at least a few separate problems should be tackled:

- Firstly, a new infrastructure has to be set up in order to be able to teach the course.
- Students will have to buy a micro and a modem, which are almost obligatory for participation.
- An on-line course is almost certain to involve the use of an electronic mail service or a conferencing system, or both.

(Schofield 1989:94)

The main thing to remember is that all choices are subject to economic constraints.

In Unisa's specific situation, the tutor systems (delivery computers) that are currently available would play a major role in the choice of hardware for authoring purposes. After careful analysis, it appears that a piggyback approach, using microcomputers, might provide the ideal situation. Because of the memory and capability range of the IBM-PCs (or compatibles), combined with their current availability, use of this hardware would provide a very economic medium. Although all computer science students should have access to a PC, in a recent study by one of our lecturers doing her M.ED on the topic of CAI, it was discovered that only about 25% of the registered students' computers have some kind of graphics facility available (or know that their computers possess graphics capabilities).

Currently we are considering using IBM PCs or 100% compatibles with at least Hercules or CGA graphics abilities - hoping that students who already possess PCs without the necessary abilities will be able to upgrade/expand their machines.

3.2.4.3 Software requirements

The search for a computer system should run parallel to the choice and evaluation of a software package. Depending on the relative importance of a specific package in the scheme of your needs, a choice of software package may dictate the choice of computer. Even though you might prefer computer A to computer B, if a software package essential for your needs will not run on A, forget it. For example, if the software package of your choice needs graphics, and the computer you prefer does not support graphics, look for another computer (Perry 1986:56; Coburn, et. al. 1982:89).

Elfring (1985:235-240) offers a three step process approach in choosing a programming language: application description, determination of programming features, and other practical considerations. By identifying the required or desired language features early, language selection becomes more than a simple comparison of features, e.g., considerations such as existing compilers and language development tools which may contribute to increased productivity as well as ease of execution. Rationale for selected language needs to address the large amounts of narrative data sets interrelated to

helps and levels typical of a tutorial approach. Additionally, user friendly language and execution along with compatibility with the selected computer (for example IBM-PC) must also be identified before the formal search begins.

In the development stage the software must be programmed in some fashion. There are basically three alternatives:

- A general purpose programming language such as BASIC, Pascal, or C;
- an authoring language such as PC/Pilot, Tutor, QAL, or TenCore;
- an authoring system such as Quest, PCD3, Wise, Prophi, or Mentor.

Most of the courseware developed to date for microcomputers has undoubtedly been programmed using some version of the BASIC language - the primary reason being not its suitability for creating instructional programs, but its simplicity and availability. Lately, programs written in some version of C or Pascal have appeared. Regardless of which of these languages are used, an explicit programming step is involved. This means that the author must code, debug and test the courseware. Furthermore, these **general-purpose languages** do not include features which make the authoring process easier - such as functions for screen display, answer analysis, response judging, student data collection, or control of multimedia devices (e.g. sound recordings and videodisk) (Kearsley 1985:213; Kontos 1984:21).

Author languages have these features because they are specifically designed for the purpose of creating CAI lessons. Major CAI systems have sophisticated author languages specially designed for these systems. Even though author languages make it easier to develop CAI courseware, they still require an explicit programming activity (Kearsley 1985:213; Kontos 1984:21).

Authoring systems, however, attempt to eliminate the need to program. By means of prompting or menu displays, the author can create courseware at a terminal without "programming" - the system automatically generates the program in fully debugged form. Since authoring systems usually have design principles built into them, the courseware developed tends to be more

instructionally sound. In addition, authoring systems can include automatic documentation capabilities which encourage authors to document their work as they develop it (Kearsley 1985:213; Kontos 1984:21).

What are the pros and cons of using general purpose-languages, author languages, and authoring systems for courseware development? Although author languages and systems were primarily designed to speed up the courseware development process and encourage transportability of courseware, they may sometimes restrict the author to the use of relatively narrow models of instruction (Kontos 1984:21). Figure 3 illustrates the major trade off involved in the three approaches between instructional flexibility and development efficiency.

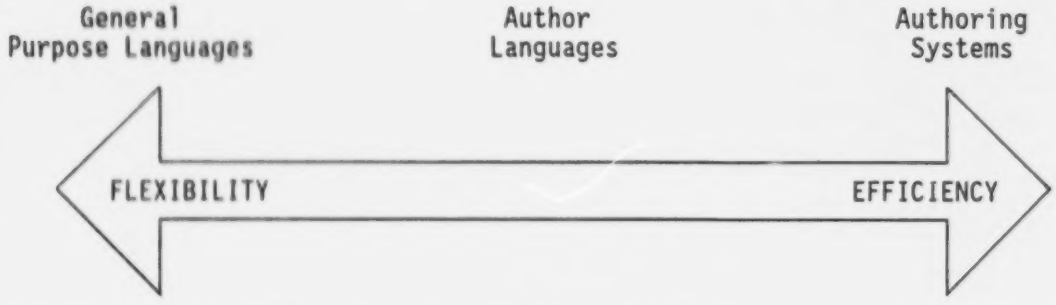


Figure 3: Trade-off between flexibility and efficiency (Kearsley 1985:215)

General-purpose languages provide maximum flexibility but require considerable programming skill and time. At the other end of the spectrum, authoring systems provide maximum efficiency but place constraints on what kind of instruction can be created. Author languages represent a compromise between flexibility and efficiency. The cost of general-purpose languages and author languages is roughly the same, while authoring systems are much more expensive.

Several requirements and "it would be nice to have's" have already been identified by members of staff working in this area. **Major criteria for choosing an authoring tool(software) for CAI in distance education** includes the following:

- **Local sales and service** - no matter how well documented, every significant piece of software generates questions from its users. When Murphy came upon a computer, he clarified his original statement, whatever can go wrong will go wrong, to say "whatever can go wrong will go wrong - and at exactly the worst possible moment".
- **Cost** - how much is an authoring tool worth? The cost factor plays a major role in choosing an authoring tool. What will the cost be for the author? What will the cost be for the student? Will the existing hardware requirements of the student change? Who would finance the project?
- **Copy protection** - the ability to make archival copies of software, which is permitted under the copyright law, is desirable and necessary in educational situations.
- **Distribution** - does the system allow for the distribution of lessons on diskettes (will the completed lessons fit on a 360K diskette - diskettes are normally upwards compatible)? Can lessons be distributed via the electronic mail network? What are the security considerations? Is there a royalty factor involved with the vendor of the software when making copies of completed lessons?
- **Buy software that does what you need it to do.** Decide upon the criteria below what your special needs are.
- **Computer-managed instruction(CMI) capabilities** - Does the tool have CMI capabilities? This is a very important aspect when working in a distance education environment. The questions to be asked are whether or not the data on the student's performance is important or not, and whether or not it should be returned to the lecturer. Can the tool provide student data (T scores, Z scores, rank ordering of students, pass/fail per objective, number of errors, number of correct, % correct, number of tries per item, etc.), have management tools for data collection, response latency measures, report programs, or monitor student progress on-line. What is the possibility that student performance data can be used as input to other software packages which performs statistical analysis.
- **Screen design and output facilities** - Does the tool have the following capabilities:

  - Text displays (editing, variety of fonts, variety of character sets);

- Graphics editor or similar capabilities (does the tool provide character sets, colour, animation, change of colour, change of character (size, colour, font), high (53200 pixels) medium or low (1600 pixels) resolution, pixel addressable positions, create basic shapes in memory, draw figures and graphs, sprites, rotate figures, store graphic images in a library for use in other lessons?);
- Lesson editor (does the tool provide immediate lesson execution mode, cursor moving commands, text changing commands (insert, delete, change attributes, reformatting, copy, replace), displaying the screen at any stage, modifying graphics and animations, etc.?);
- Test editor (allowing for multiple choice, fill in, dichotomous, listing, labelling, identifying, matching and short answer questions, random question selection from question pool, allocating weighed values to questions);
- Text and graphics from other computer packages (does the tool allow the transportation of text created on a word processor or graphics created by an advanced graphics package into the lessons?);
- Peripheral interfacing (does the tool allow access to special features such as interactive video tape or disk, slides, sound recordings, synthetic speech, paddles, joy stick, touch screen, light pen, graphics tablet, mouse, calculation pad, shell, etc.?);
- Answer-matching capabilities (does the tool have the necessary command structures to handle student responses - does the tool provide a spelling tolerance, acknowledgement of inappropriate spaces, changes in upper and lower case letters, matching synonyms, provision of alternate answers, etc.?);
- Branching capabilities (has the tool branching capabilities to cater the different methods of programmed instruction - linear, direct, dynamic, looping, random, to other lessons?);

- **Instructional design features** - the importance of this feature varies according to the configuration of the authoring team chosen. This feature is important where you have a great number of lecturers which are subject experts but do not have an educational background - it is in this case necessary to choose an authoring tool that gives at least some assistance in this regard, without limiting the freedom of the lecturer to try another approach as well.

- **Hardware requirements** - what hardware is required for developing lessons and for running the completed lessons? Will the existing available hardware be suitable/adequate, if not, can the existing hardware be upgraded or modified without great expense? What about machine incompatibility - will programs developed on one type of machine run on any other brand of machine? Which operating system is needed to run the programs - does the hardware support this type of operating system?
- **Support documentation** - does the software have adequate documentation? Does the documentation provide understandable directions, the purpose of the software, a table of contents, how to execute the software (including hardware requirements and handling of peripherals), and a documentation index with accurate pagination (absolutely essential)? Does the documentation facilitate rapid use of the tool?
- **Link to other computer software, general purpose computation** (a high level language, or other computer package). Facilities beyond the scope of the authoring tool are sometimes required. It is therefore desirable to have an interface between the authoring tool and a conventional programming language or the operating system - especially in the case of computer science students using such packages in laboratory sessions. Furthermore the ability to add features which are needed for a specific application to the existing authoring tool would be desirable - one such requirement that we have identified is the ability to transfer parameter values from the authoring tool to a program written in another high level language, and vice versa.
- Ease of learning - how easy is it for the novice to master the authoring tool? Are training courses in the use of the authoring tool available - maybe a CAI tutorial, or a workshop offered by the vendor?
- Ease of use - how easy is it to write interactive CAI after you have learned the primary commands of the authoring tool? Do you see the same actual screen the student will use (what you see is what you get)? Are menus and prompt lines available? Are help screens available while you are busy authoring? Is it easy to test or evaluate courseware while you are busy developing it? Can the tool be used by the novice CAI developer? Does the tool have a variety of program utilities, such as listing programming steps, printing of screens/frames, editing of existing frames?

- **Overall performance** - what is the overall impression that the final product maintains. How fast does it react to some kind of input from the student, are there long pauses of nothing happening when branching between frames, etc.

(Perry 1986:56-58; Coburn et.al 1982: 89-91; Grabinger 1985:21-23; Gillingham, Murphy, Cresci, Klevenow, Sims-Tucker, Slade & Wizer 1986:7-17; De Villiers 1988:9-13)

Guidelines of when to use an authoring system and when to use a general purpose programming language.

Depending on your programming experience level:

- Select a good, suitable and reliable authoring system and use it as far as possible (why re-invent the wheel when time is precious?).
- When you need some kind of programming that your authoring system cannot handle, try to program it in the authoring language that accompanies all high-quality authoring systems. This programming code may often be called directly from your authoring system.
- However, if your authoring language does not support the solving of the problem you are experiencing, only then should you program it in a general purpose programming language.

"A good program lets you concentrate on your task, not on the program." (Grabinger 1985:23)

3.3 TUTOR SYSTEM

The tutor system acts as a tutor to the student. It includes the following components:

- Presentation - shows the contents of a course to a student.
- Tutoring - leads the student through a course with many learning-supporting functions.
- Laboratory - enables the "trying-out" of the currently learned material (Schlageter et. al. 1988:11).

The same guidelines for choosing the hardware and software for the author system should be followed in choosing the tutor system - in fact, the two must be done simultaneously.

To be able to access and use the CAI system via a communication network, a student needs the following equipment:

- A computer (IBM-PC or compatible fitting the hardware requirements defined at the author system);
- a telephone and modem for receiving and sending digital information by telephone lines;
- a communication software package at the user end; and
- access to a data communication network.

(Charp 1988:162; Schofield 1989:94)

In this case the applicable CAI courseware is downloaded from the course database via the communication network onto the students computer, from where the student works through the actual study material. If any results have to be returned to the course database, the opposite route has to be followed.

Use of an electronic mail network can, however, not be the only option available to our students as many of them do not have a telephone at their disposal - even if they have access to a computer. This does not only apply to rural areas but also to students living in new or fairly new urban areas (where waiting up to five years to obtain a private telephone is nothing new).

In the last case, the use of diskettes that are distributed via the postal network is the only viable option available. Since this infrastructure for the distribution of diskettes already exists, the only problem would be the duplication of such diskettes and the use of diskettes that are compatible with the student's specific computer configuration.

3.4 COMMUNICATION SUPPORT

This module supports the use of a connected communication network. There are two main applications:

- Distributed learning system - as the student work-stations are geographically distributed all over South Africa, the access to the course data base must be realized efficiently and at low cost (via some kind of modem).
- Dialogue/mailing - messages can be sent to other students or to the university in a convenient way.

A major problem of a remote teaching system is the lack of communication facilities between students or between students and the university. At a conventional university, students work mostly in groups. They learn to solve problems or develop projects by doing teamwork. The possibility of discussing problems with the professor or assistant is a very important part of the study.

This disadvantage can be solved in an electronic learning system by integrating comfortable mailing functions (Schlageter et al. 1988:12).

A system is needed which allows mail to be sent to specified groups of receivers (students or mentors) which can be defined by the sender of the message. For example, a message may be sent to a specified student, the mentor of a specified course, all students who are currently participating in a specified course, all students living close to a regional office, etc.

In addition, the communication facilities should contain a means of downloading the latest version of appropriate courseware software to the student's computer at a low cost.

At the moment, videotex (Beltel from the POTS) connected to SAPONET seems to be a possible database to store and distribute courseware. There is a program available called PC Belnet which allows one to download and upload files/clusters of files onto Beltel. It will, however, become quite

expensive in terms of telephone time for students living some distance from the nearest Beltel "station".

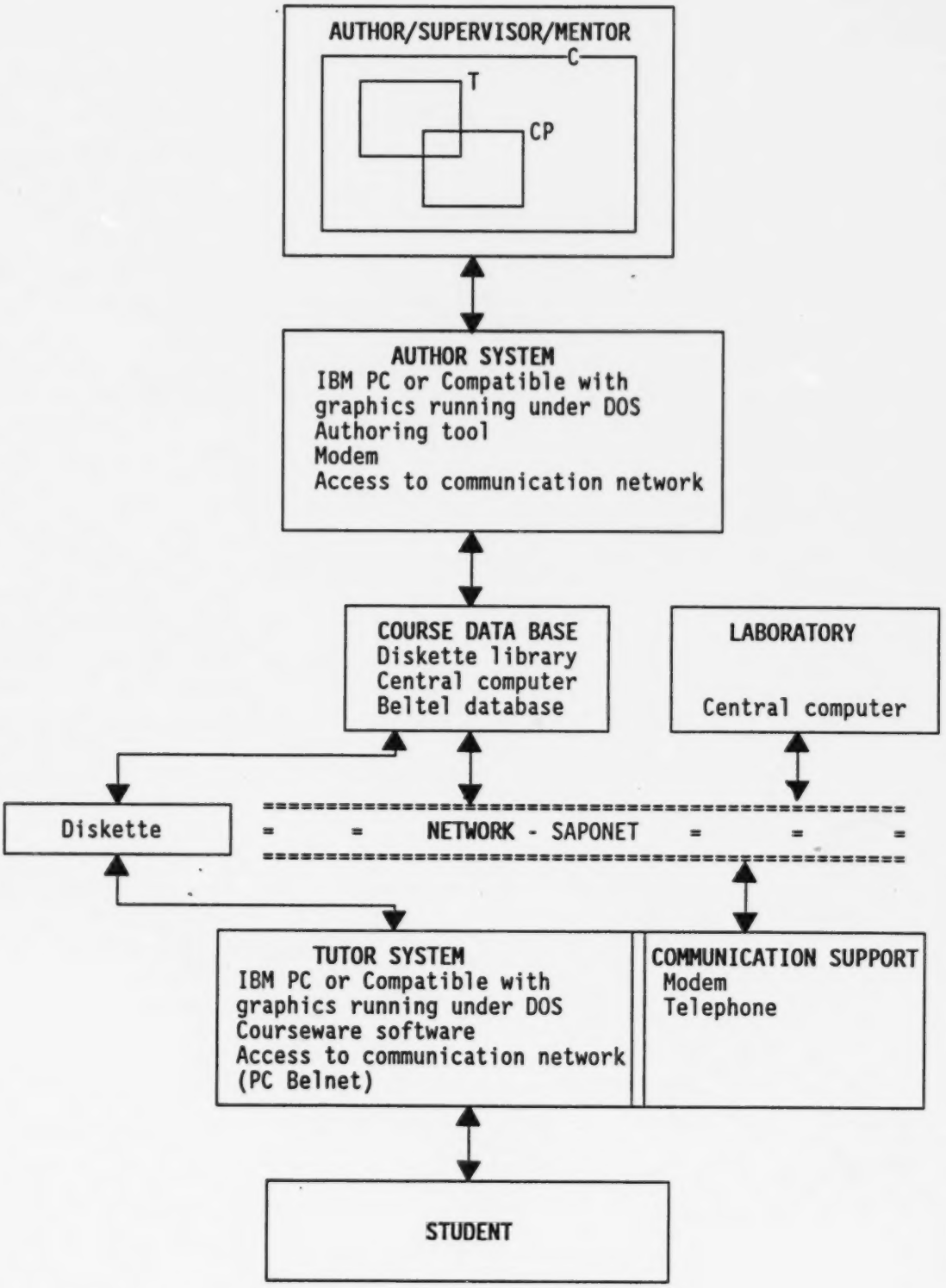
Another possibility is storing the courseware on a central computer at the university from which students have to collect what courseware is appropriate via a modem. Access during peak time could, however, course a problem.

In addition, efficient exercises for courses on programming, compilers, data bases, expert systems, etc. usually require large-scale software tools which cannot be produced for each course and be integrated into it individually (mainly due to copyright restrictions).

There are three main approaches to how an existing tool can be used for an electronic course - it can either be used separately, loosely coupled with the courseware, or fully integrated into the courseware. In the first approach, the student himself has to switch explicitly between the courseware and the tool, the details depending on the student's computer, the underlying operating system, etc. In the worst case, he may even have to struggle with several floppy disks and software he had to obtain (buy) himself; in the best case, he may merely have to switch between two concurrent processes (limited, however, by the memory size of his computer). The last approach usually cannot be realized without significant changes to existing tools as well as courseware, and thus for the time being seems unrealistic (Schlageter 1988:13-17).

The use of public domain software tools that are freely available at a very low cost (if any) and that can be downloaded to the student's own computer would be the ideal solution for use in the loosely coupled or separate situation. However, the possibility that such software tools are not available, and originals might have to be used, should also be considered. This can possibly be accomplished by "borrowing" the tool from the central computer by means of a network (remote login). This means that the software tool will actually be shared by all students requiring that tool, which may lead to problems in accessing a tool at the stage the student needs it (especially around due dates of assignments). Using the network in this way will be researched to expose the possibilities and restrictions thereof.

3.5 SUMMARY



30

4. CONCLUSION

Because CAI participants have different purposes, backgrounds and experiences, they require different kinds of help, simulations and tutorials. Because it results in effective participant learning experiences, this individualized instruction capability provides the primary rationale for computer-based education/training.

Yet this individualized instruction can be very costly. Very few people may want to approach the huge investment involved in building CAI from scratch, simply because it takes approximately 200 man-hours to produce a single hour of deliverable CAI. Therefore, this output hour must be time well spent, resulting in enhancements to quality and productivity. Unfortunately, this payback may take several years to become apparent.

It is this very investment, this individualized treatment which the participant receives, that can contribute to CAI success. The learner should be allowed to make choices affecting the pace, sequence and level of the presentation. This capacity for interaction permits information delivery, tailored to individual users, through a careful blend of systems controls and direct user control. Interactive conversational exchange is essential to success.

CAI also provides for easy incorporation of E.L. Thorndike's major laws of learning: Exercise, theory and repetition, and effect. In a typical tutorial/drill and practice program, the computer presents the student with a few pages of information followed by a short quiz. Reinforcement of material occurs, either through satisfactory results which strengthen the bond between the encountered situation and the correct response, or between the encountered situation and the remedial response. But more important is the instantaneous, non-threatening feedback in the form of a computer response (which the normal written drill cannot provide). It is then that the CAI computer is acting almost like a human tutor - making it a much more enjoyable and motivating experience.

Mark Twain once said that the best form of education is one teacher sitting at the end of a log and a schoolboy on the other. Maybe we could change this to read that the best form of education is one teacher, authoring quality lessons, sitting at the end of a CAI system and a schoolboy doing the lessons on the other. Material presented by means of CAI replaces the current paper-based tuition method with a one-to-one type of interaction which supports individual activities of various kinds - this may even be preferable to the one-to-many interaction in the conventional classroom tuition.

Despite its relative values, the feeling is that CAI should not be used in an examination environment, simply because findings support the learning style theory that students who are visually perceptive, achieve more when using a visual individualized learning tool than those who are not visually perceptive. Study results indicate that CAI is not a means of individualized instruction for all students - a mismatch of learning style and learning tool can be a spanner in the works of the learning process. Additionally, testing can result in a deterioration in the non-threatening, reinforcement atmosphere essential to CAI topic absorption and subsequent retention. Furthermore, CAI is not a suitable medium for all subjects. For example, teaching motor skills (other than for typing) needs other types of media.

A final word of warning: Although the implementation of CAI in distance education seems to be the solution to many problems, it will only be as successful as the quality of courseware that is produced (for distant education it has to be very good, well-documented and **foolproof**).

Perhaps this can all be summed up in the following words: "**The driving force of innovating remote education (and education in general) is not 'hardware' or 'software (or course meaning teaching material)' but 'heartware' - education is not a business but a social duty**" (Shiba, Matsuda & Takano 1988:54).

BIBLIOGRAPHY

BETLACH JA. 1986. **Framework for Computer Assisted Instruction Courseware: A Case Study.** Journal of Educational Technology Systems, vol. 15, no.1.

BOON KL, & DE WOLF HC. 1988. **Tele-Education: An Experiment in the Netherlands.** In: Remote Education and Informatics: Teleteaching, edited by E. Lovis, Amsterdam: North-Holland, 1988.

BURKE RV. 1982. **If You're Thinking About Developing Your Own Software, Consider This...** Electronic Education, December 1982.

CHARP S. 1988. **Telecommunications as an asset in Education.** In: Remote Education and Informatics: Teleteaching, Amsterdam: North-Holland, 1988.

COATES B. 1982. **Curriculum Development in CBL and the Impact on Teaching Styles and Methodology.** Computers and Education, no. 41, 1982.

COBURN P, KELMAN P, ROBERTS N, SNYDER T, WATT D & WEINER C. 1982. Practical Guide to Computers in Education, Reading, Massachusetts: Addison-Wesley.

DE VILLIERS C. 1988. Outeurstelsels. Tegniese verslag TR/88-04, Departement Rekenaarwetenskap en Inligtingstelsels, Unisa. Pretoria: Unisa.

DAVIES G, & PREECE J. 1988. **The Use of Distance Learning to Provide Proffesional Retraining in Computing.** In: Remote Education and Informatics: Teleteaching, edited by E. Lovis, Amsterdam: North-Holland, 1988

ELFRING G. 1985. **Choosing a Programming language.** Byte, June 1984

GILLINGHAM M, MURPHY P, CRESCI K, KLEVENOW S, SIMS-TUCKER B, SLADE D & WIZER D. 1986. **An Evaluation of Computer Courseware Authoring Tools and a Corresponding Assessment Instrument for Use by Instructors.** Educational Technology, September 1986.

GLEASON G. 1981. Microcomputers in Education: The State of the art. Educational Technology, March 1981.

GRABINGER RS. 1985. An Evaluation of Three Authoring-Language Software Packages. TechTrends, vol. 30, no. 4, May-June 1985.

HOLMBERG B. 1978. Distance Education - a survey and bibliography. Kogan Page Ltd.

KEARSLEY G. 1985. Microcomputer Software: Design and Development Principles. Journal of Educational Computing Research, vol. 1 no. 2.

KONTOS G. 1984. Authoring Quality Software for Education. Computer Education, June 1984.

KONTOS G. 1985. Instructional Computing: In Search of Better Methods for the Production of CAI Lessons. Computer Education, February 1985.

KOTZE P. 1988. Onderwystegnologie - Geprogrammeerde Onderrig en Rekenaarondersteunde Onderrig in Perspektief. Tegniese verslag TR/88-05, Departement Rekenaarwetenskap en Inligtingstelsels, Unisa. Pretoria: Unisa.

MAURER H, & MAKEDON F. 1988. COSTOC: Computer Supported Teaching Of Computer Science. In: Remote Education and Informatics: Teleteaching, edited by E. Lovis, Amsterdam: North-Holland, 1988.

MAKEDON F, & MAURER H. 1988. CLEAR: Computer LEArning Resource Centers. In: Remote Education and Informatics: Teleteaching, edited by E. Lovis, Amsterdam: North-Holland, 1988.

OTTMAN T. 1988. Can Teaching by Computer Replace Teaching by Professors?. In: Remote Education and Informatics: Teleteaching, edited by E. Lovis, Amsterdam: North-Holland, 1988.

PERRY WE. 1986. How to Test Software Packages. New York: John Wiley & Sons.

PRESSMAN I & ROSENBLOOM B. 1978. CAI: Its Cost and Its Role. Journal of Educational Technology Systems, vol. 6, no. 2.

PRICE RV. 1982. Selecting Educational Computer Software. Educational Resources and Techniques, Spring/Summer 1982.

SCHLAGETER G, SIX H-W, STERN W, & UNGER C. 1988. Remote Education with Online Communication and Laboratories. In: Remote Education and Informatics: Teleteaching, edited by E. Lovis, Amsterdam: North-Holland, 1988.

SCHOFIELD J. 1989. Negotiating the IT learning curve. Practical Computing, April 1989.

SHIBA S, MATSUDA T & TAKANO H. 1988. Role of Educational Industry in Teleteaching - A Case Study in Japan: KUMON. In: Remote Education and Informatics: Teleteaching, edited by E. Lovis, Amsterdam: North-Holland, 1988.

WATSON DM. 1983. A Model for the Production of CAL Material. Computers and Education, vol.7 no. 3.